# DragonFlyBSD - Bug #1720

## Patch: header file from FreeBSD current used everywhere in drm

04/09/2010 04:31 AM - davshao

| | | | |
|---|---|---|---|
| **Status:** | Closed | **Start date:** | |
| **Priority:** | Normal | **Due date:** | |
| **Assignee:** | | **% Done:** | 0% |
| **Category:** | | **Estimated time:** | 0.00 hour |
| **Target version:** | | | |

**Description**

The following patch is the first part of porting the lastest version
of graphics drm from FreeBSD Current, porting that is eventually able
to run a Radeon HD 4550 using hardware accelerated drivers and not the
default soft fallback of Mesa.

The file patched is a common header file used everywhere in drm. The
vm/vm_page.h appears to be used by the via drivers which I have not as
yet added--they are completely new. But this shows how features from
individual drivers influence the overall common code. The unlikely
definition is used by what will later be added as a new file drm_mm.c.

What is more interesting is what is not included in the patch that is
in FreeBSD Current. One part not included is a
SYSCTL_DECL(_hw_drm);
that appears related to msi interrupts in drm_drv.c
SYSCTL_INT(_hw_drm, OID_AUTO, msi, CTLFLAG_RDTUN, $drm_msi, 1,
"Enable MSI interrupts for drm devices");

DragonFly has a related section #ifdef'd to 0 so it seemed unwise to
enable this.

Secondly in drmP.h, FreeBSD defines a prototype
void drm_close(void * data)
that is passed as a function pointer in drm_fops.c and that has a
slightly different prototype from DragonFly's version in drm_drv.c
But DragonFly's code in drm_fops.c does not need drm_close as a
function pointer at all.

Third and most important for future porting, FreeBSD has in drmP.h
struct unrhdr *drw_unrhdr.

What FreeBSD appears to have done is to come up with a partial but not
complete solution to fitting the semantics of Linux's idr (small
integer ID management) API, the API that is the biggest barrier to
porting GEM. What Intel has been doing is to develop pseudo-file
descriptors that associate small integers with blobs of data. They
would have used real file descriptors if the OS's had not had
relatively small limits on file descriptor numbers. It appears that
what is needed is an API where a small integer greater than some floor
is returned. What I am hoping to prove very shortly is that these
semantics can, perhaps with terrible performance, be emulated using
the red-black tree that currently backs the blob storage. Eventually
the BSDs should probably settle on a common implementation using radix
trees which seem suited for retrieving either smallest available
integers or smallest available integers greater than some floor
relatively quickly.

```
diff --git a/sys/dev/drm/drmP.h b/sys/dev/drm/drmP.h
index 3f98d20..3ca6fb2 100644
--- a/sys/dev/drm/drmP.h
+++ b/sys/dev/drm/drmP.h
@@ -63,6 +63,8 @@ struct drm_file;
```

```
 #include <vm/pmap.h>
 #include <vm/vm_extern.h>
 #include <vm/vm_map.h>
+#include <vm/vm_object.h>
+#include <vm/vm_page.h>
 #include <vm/vm_param.h>
 #include <machine/param.h>
 #include <machine/pmap.h>
@@ -141,6 +143,8 @@ MALLOC_DECLARE(DRM_MEM_AGPLISTS);
 MALLOC_DECLARE(DRM_MEM_CTXBITMAP);
 MALLOC_DECLARE(DRM_MEM_SGLISTS);
 MALLOC_DECLARE(DRM_MEM_DRAWABLE);
+MALLOC_DECLARE(DRM_MEM_MM);
+MALLOC_DECLARE(DRM_MEM_HASHTAB);

 #define DRM_MAX_CTXBITMAP (PAGE_SIZE * 8)

@@ -185,6 +189,11 @@ typedef void            irqreturn_t;
 #define IRQ_HANDLED       /* nothing */
 #define IRQ_NONE          /* nothing */

+#define unlikely(x)             __builtin_expect(!!(x), 0)
+#define container_of(ptr, type, member) ({              \
+    __typeof( ((type *)0)->member ) *__mptr = (ptr);    \
+    (type *)( (char *)__mptr - offsetof(type,member) );})
+
 enum {
     DRM_IS_NOT_AGP,
     DRM_IS_AGP,
```

### History

**#1 - 01/19/2012 05:00 PM - swildner**

*- Description updated*

*- Status changed from New to Closed*

*- Assignee deleted (0)*

It was committed in 3f6063cc01b519b28ab36c05951d44c32dbf6a51

Thanks!